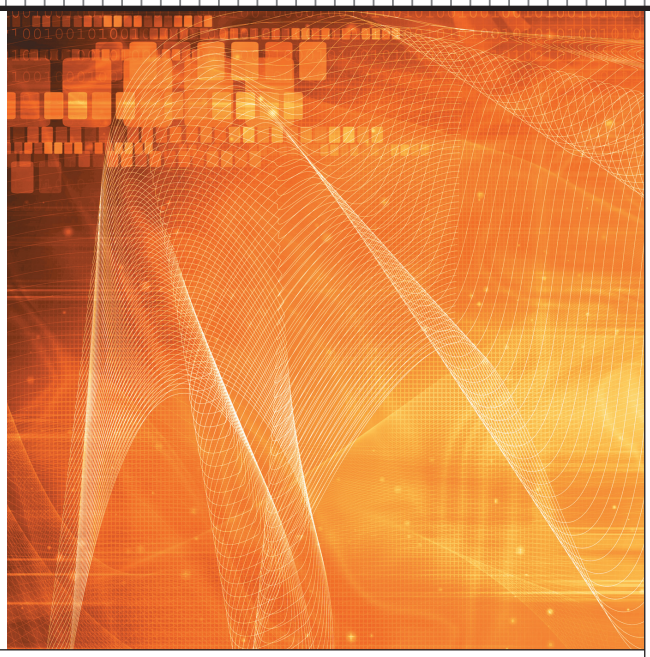
Adding Attributes to Role-Based Access Control

- **D. Richard Kuhn,** *National Institute of Standards and Technology*
- **Edward J. Coyne,** *Science Applications International Corporation*
- **Timothy R. Weil,** *Raytheon Polar Services Company*



Merging the best features of RBAC and attribute-based systems can provide effective access control for distributed and rapidly changing applications.

Role-based access control (D.F. Ferraiolo and D.R. Kuhn, "Role-Based Access Controls," *Proc. 15th Ann. Nat'l Computer Security Conf.,* NSA/NIST, 1992, pp. 554-563; R. Sandhu et al., "Role-Based Access Control Models," *Computer*, Feb. 1996, pp. 38-47), also known as RBAC, provides a popular model for information security that helps reduce the complexity of security administration and supports review of permissions assigned to users. This feature is critical to organizations that must determine their risk exposure from employee IT system access.

RBAC has frequently been criticized for the difficulty of setting up an initial role structure and for inflexibility in rapidly changing domains. A pure RBAC solution may provide inadequate support for dynamic attributes such as time of day, which might need to be considered when determining user permissions. To support dynamic attributes, particularly in large organizations, a "role explosion" can result in thousands of separate roles being fashioned for different collections of permissions. Recent interest in attribute-based access control (ABAC) suggests that attributes and rules could either replace RBAC or make it more simple and flexible.

ROLE-BASED ACCESS CONTROL

A US standard defined in *ANSI/INCITS 359-2004, Information Technology—Role Based Access Control,* RBAC controls all access through roles assigned to users. Each role assigns a collection of permissions to users. RBAC assumes that, in most applications, permissions needed for an organization's roles change slowly over time, but users may enter, leave, and change roles rapidly.

For efficiency, roles can be structured hierarchically so that some roles inherit permissions from others. RBAC simplifies access control compared with the administrative burden that would be required for a direct mapping from individual users to access control lists attached to resources. Once roles with their permissions have been defined, user provisioning simply requires that office staff assign users to roles as authorized by management.

RBAC is also well suited to separation-of-duty requirements, where no single individual has all permissions needed for critical operations such as expenditure of funds. Proper operation of RBAC requires that roles fall under a single administrative domain or have a consistent definition across multiple domains, so distributed applications might be challenging.

Although RBAC implementations differ, many provide at least basic features of the RBAC standard. Several proposals for revising the standard have been introduced (N. Li, J. Byun, and E. Bertino, "A Critique of the ANSI Standard on Role-Based Access Control," *IEEE Security & Privacy*, Nov. 2007, pp. 41-49) and evaluated with respect to the rationale for design decisions in the current standard (D.F. Ferraiolo,

0018-9162/10/$26.00 © 2010 IEEE Published by the IEEE Computer Society

D.R. Kuhn, and R. Sandhu, "RBAC Standard Rationale: Comments on a Critique of the ANSI Standard on Role-Based Access Control," *IEEE Security & Privacy*, Nov. 2007, pp. 51-53).

Committee CS1.1 within the International Committee for Information Technology Standards (INCITS) has initiated a revision with the goal of extending its usefulness to more domains, particularly distributed applications.

ATTRIBUTE-BASED ACCESS CONTROL

Although ABAC has no clear consensus model to date, the approach's central idea asserts that access can be determined based on various attributes presented by a subject (A.H. Karp, H. Haury, and M.H. Davis, "From ABAC to ZBAC: the Evolution of Access Control Models," tech. report HPL-2009-30, HP Labs, 21 Feb. 2009). Rules specify conditions under which access is granted or denied. For example, a bank might allow access if the subject is a teller working between the hours of 7:30 am and 5:00 pm, or the subject is a supervisor or auditor working those same hours who also has management authorization.

This approach might be more flexible than RBAC because it does not require separate roles for relevant sets of subject attributes, and rules can be implemented quickly to accommodate changing needs. The trade-off for this flexibility is the complexity of cases that must be considered: for n Boolean attributes or conditions using attributes, there are 2^n possible combinations. Authentication of attributes could be distributed and based on the authority that issues a particular attribute, such as a firm vouching for a subject's employment status. Negotiation between parties must establish trust in attributes and ensure that parties use the same definition for attributes.

Table 1. Combination strategies and options for integrating attributes with RBAC.

Option	*U*	*R*	*A*	Model	Permission mapping
0	0	0	0	undefined	—
1	0	0	1	ABAC-basic	$A_1, \ldots, A_n \rightarrow$ perm
2	0	1	0	undefined	—
3	0	1	1	ABAC-RBAC hybrid	$R, A_1, \ldots, A_n \rightarrow$ perm
4	1	0	0	ACLs	$U \rightarrow$ perm
5	1	0	1	ABAC-ID	$U, A_1, \ldots, A_n \rightarrow$ perm
6	1	1	0	RBAC-basic	$U \rightarrow R \rightarrow$ perm
7	1	1	1	RBAC-A, dynamic roles	$U, A_1, \ldots, A_n \rightarrow R \rightarrow$ perm
8	1	1	1	RBAC-A, attribute-centric	$U, R, A_1, \ldots, A_n \rightarrow$ perm
9	1	1	1	RBAC-A, role-centric	$U, R, A_1, \ldots, A_n \rightarrow$ perm

* ***U* = user/subject ID; *R* = role; *A* = attributes**

COMBINING RBAC AND ABAC

RBAC and ABAC have their particular advantages and disadvantages. While ABAC may require up to 2^n rules for n attributes, attempting to implement the same controls in RBAC could, in a worst case, require 2^n roles, one for each possible combination of attributes. Generally speaking, RBAC trades up-front role structuring effort for ease of administration and user permission review, while ABAC makes the reverse trade-off: it is easy to set up, but analyzing or changing user permissions can be problematic.

Determining RBAC role structure, a process known as *role engineering* (E.J. Coyne, *Role Engineering*, Artech House, 2008), could take many months, but once completed it is easy to determine who has what permissions. ABAC makes it easy to specify access rules, but to determine the permissions available to a particular user a potentially large set of rules might need to be executed in exactly the same order in which the system applies them. This can make it impossible to determine risk exposure for a given employee position.

Can these two models be combined to take advantage of both their strengths?

Table 1 summarizes possible combination strategies and options for integrating attributes with RBAC (RBAC-A). Options 0 and 2 are undefined but included for completeness; options 1 and 3, which have no user ID in the access decision, might appear unusual but could be used in public facilities where attributes or roles determine anonymous users' access.

Broadly speaking, there are three RBAC-A approaches to handle the relationship between roles and attributes, all retaining some of the administrative and user permission review advantages of RBAC while allowing the access control system to work in a rapidly changing environment:

- *Dynamic roles*. Attributes such as time of day are used by a front-end module to determine the subject's role, retaining a conventional role structure but changing role sets dynamically (R. Fernandez, *Enterprise Dynamic Access Control Version 2 Overview,* US Space and Naval Warfare Systems Center, 1 Jan. 2006; http://csrc.nist.gov/rbac/EDACv2overview.pdf). Some implementations of dynamic roles might let the user's role be fully determined by the front-end attribute engine, while others might use the front end only to select from among a predetermined set of authorized roles.

- *Attribute-centric.* A role name is just one of many attributes. In contrast with conventional RBAC, the role is not a collection of permissions but the name of an attribute called "role." This approach's main drawback is the rapid loss of RBAC's administrative simplicity as more attributes are added. It also suffers from potential problems with ABAC when determining the risk exposure of a particular employee position.
- *Role-centric.* Attributes are added to constrain RBAC. Constraint rules that incorporate attributes can only reduce permissions available to the user, not expand them. Some of ABAC's flexibility is lost because permission sets are still constrained by role, but the system retains the RBAC capability to determine the maximum set of user-obtainable permissions. As an aside, developers explicitly designed the formal model for RBAC, introduced in 1992, to accommodate additional constraints being placed on a role.

The dynamic-roles RBAC-A model allows implementation as a layer atop an existing RBAC structure. Attribute-centric RBAC-A, as defined here, is less a true RBAC system because access is not controlled by roles formed from sets of permissions.

PERMISSION CONSTRAINTS

Implementing the role-centric RBAC-A scheme requires changing the RBAC standard to constrain the set of permissions available during a user's session. In the current standard, permissions are available depending on the user's active roles. Clearly, the subject must avoid receiving any permission not authorized for the active role or restricted by the attribute-based constraints.

The permissions in this approach will be the intersection of P and R, where P is the set of permissions assigned to the subject's active roles and R is the set of permissions specified by the applicable ABAC rules. The user's role set therefore determines the maximum set of available permissions, supporting the principle of least privilege and allowing easy review of user permissions.

Combined design

We use a combined design rather than a pure system because, in general, some user attributes are relatively static—such as position, skill set, or office location—while others, such as time of day, are dynamic. Developing a role structure based on the more static attributes can avoid awkward designs that might result from purely one choice or another.

For example, consider a system with 10 attributes, four of which are static and six dynamic. This set of attributes could result in 2^{10} roles or 2^{10} ABAC rules. Establishing a role structure based on the four static and six dynamic attributes means a maximum of 16 roles and 64 rules, a significant improvement over the 1,024 roles or rules that could be considered using only RBAC or only ABAC.

Determining maximum permissions

Applying the role-engineering effort to the relatively static attributes, and encapsulating these components of access decisions in roles, can reduce the number of dynamic rules dramatically. A combined design thus retains advantages of RBAC, such as ease of user provisioning and the ability to quickly determine the maximum permissions available to each user—critical in determining risk exposure while preventing a "role explosion" to cover every possible contingency for permission sets that might be required by users.

We believe this is an appropriate trade-off that will retain the benefits of RBAC while extending its utility to today's important distributed applications.

In response to comments received over the five-year life of the current RBAC standard, INCITS CS1.1 is developing a policy-enhanced RBAC standard to accommodate importation of arbitrary constraints, including attributes of all types. This enhanced model will maintain the advantages of RBAC while providing a mechanism for including attributes in access-control decisions. For more information, see http://csrc.nist.gov/groups/SNS/rbac/rbac-standard-revision.html.

D. Richard Kuhn *is a computer scientist at the National Institute of Standards and Technology. Contact him at kuhn@nist.gov.*

Edward J. Coyne *is a senior security engineer at Science Applications International Corporation. Contact him at ed.coyne@va.gov.*

Timothy R. Weil *is an information security manager at Raytheon Polar Services Company. Contact him at timothy.weil.contractor@usap.gov.*

Disclaimer

Certain software products are identified in this document, but such identification does not imply recommendation by the US National Institute of Standards and Technology or other agencies of the US government, nor does it imply that the products identified are necessarily the best available for the purpose.

Editor: Jeffrey Voas, National Institute of Standards and Technology; j.voas@ieee.org